CLAIMS

What is claimed is:

1. A method for branching during programmable processing in a computer graphics pipeline, comprising:

(a) receiving data;

(b) performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set;

(c) branching between the programmable operations in a programmable manner; and

(d) storing the output in memory.

2. The method as recited in claim 1, wherein the programmable operations are branched to labels.

3. The method as recited in claim 2, wherein the labels are stored in a table.

4. The method as recited in claim 3, wherein the programmable operations are branched to indexes in the table.

5. The method as recited in claim 4, wherein each index is stored in an address register.

6. The method as recited in claim 2, wherein each index is calculated.

7. The method as recited in claim 1, and further comprising terminating the programmable operations after a predetermined number of operations have been performed.

8. The method as recited in claim 1, wherein the programmable operations are branched based on condition codes.

9. The method as recited in claim 8, wherein the condition codes are sourced as EQ(equal), NE(not equal), LT(less), GE(greater or equal), LE(less or equal), GT(greater), FL(false), and TR(true).

10. The method as recited in claim 8, wherein the condition codes are maskable.

11. The method as recited in claim 8, wherein the condition codes are swizzled.

12. The method as recited in claim 1, wherein the operations are selected from the group consisting of a branch operation, a call operation, a return operation, a cosine operation, a sine operation, a floor operation, a fraction operation, a set-on-equal-to operation, a set false operation, a set-on-greater-than, a set-on-less-than-or-equal operation, a set-on-not-equal-to operation, a set true operation, a no operation, address register load, move, multiply, addition, multiply and addition, reciprocal, reciprocal square root, three component dot product, four component dot product, distance vector, minimum, maximum, set on less than, set on greater or equal than, exponential base two (2), logarithm base two (2), exponential, logarithm, and/or light coefficients.

13. A computer program product for branching during programmable processing in a computer graphics pipeline, comprising:

(a) computer code for receiving data;

(b) computer code for performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set;

(c) computer code for branching between the programmable operations in a programmable manner; and

(d) computer code for storing the output in memory.

14. A system for branching during programmable vertex processing, comprising:
(a) a source buffer for storing data;
(b) a functional module coupled to the source buffer for performing programmable operations on the data received therefrom in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set; and
(c) a register coupled to the functional module for storing the output;
(d) wherein functional module is capable of branching between the programmable operations.

15. A method for programmable processing with condition codes in a computer graphics pipeline, comprising:
(a) receiving data;
(b) performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set including condition codes; and
(c) storing the output in memory.

16. The method as recited in claim 15, wherein the condition codes are utilized to move between the programmable operations.

17. The method as recited in claim 15, wherein the condition codes are utilized to control write masks.

18. The method as recited in claim 17, wherein the write masks are controlled utilizing an AND operation.

19. A computer program product for programmable processing with condition codes in a computer graphics pipeline, comprising:

(a) computer code for receiving data;

(b) computer code for performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set including condition codes; and

(c) computer code for storing the output in memory.

20. A system for programmable vertex processing with condition codes, comprising:

(a) a source buffer for storing data;

(b) a functional module coupled to the source buffer for performing programmable operations on the data received therefrom in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set including condition codes; and

(c) a register coupled to the functional module for storing the output.

21. A method for programmable processing in a computer graphics pipeline utilizing a predetermined instruction set, comprising:

(a) receiving data from a source buffer;

(b) performing programmable operations on the data in order to generate output, wherein the operations are programmable by a user utilizing instructions from a predetermined instruction set; and

(c) storing the output in a register;

(d) wherein the operations include at least ten (10) operations selected from the group consisting of a branch operation, a call operation, a return operation, a cosine operation, a sine operation, a floor operation, a fraction operation, a set-on-equal-to operation, a set false operation, a set-on-greater-than, a set-on-less-than-or-equal operation, a set-on-not-equal-to operation, a set true operation, a no operation, address register load, move, multiply, addition, multiply and addition, reciprocal, reciprocal square root, three component dot

product, four component dot product, distance vector, minimum, maximum, set on less than, set on greater or equal than, exponential base two (2), logarithm base two (2), exponential, logarithm, and/or light coefficients.

22. A method for executing a function in a computer graphics pipeline, comprising:

(a) receiving input data in a computer graphics pipeline;

(b) directly performing a mathematical function on the input data in order to generate output data, wherein the mathematical function is directly performed in the computer graphics pipeline; and

(c) storing the output data in memory on the computer graphics pipeline.

23. The method as recited in claim 22, wherein the mathematical function is selected from the group consisting of sine, cosine, exponent base two (2), and logarithm base two (2).

24. The method as recited in claim 22, wherein the mathematical function is selected from the group consisting of sine, cosine, tangent, arctangent, exponent, logarithm, antilogarithm, hyperbolic sine, hyperbolic cosine, hyperbolic tangent, and hyperbolic arctangent.

25. The method as recited in claim 22, wherein the mathematical function is a function in which an initial *n* derivatives are capable of being tabulated and accessed via an interpolation operation.

26. The method as recited in claim 22, wherein the input data is in a floating-point format.

27. The method as recited in claim 22, wherein the mathematical function is performed utilizing a Taylor Series.

28. The method as recited in claim 22, wherein the mathematical function is performed utilizing a cordic algorithm.

29. The method as recited in claim 22, and further comprising converting the input data from a first coordinate system to a second coordinate system.

30. The method as recited in claim 22, wherein the mathematical function is carried out in one cycle in the computer graphics pipeline.

31. A computer program product for executing a function in a computer graphics pipeline, comprising:

(a) computer code for receiving input data in a computer graphics pipeline;

(b) computer code for directly performing a mathematical function on the input data in order to generate output data, wherein the mathematical function is directly performed in the computer graphics pipeline; and

(c) computer code for storing the output data in memory on the computer graphics pipeline.

32. A system for executing a function in a computer graphics pipeline, comprising:

(a) a computer graphics pipeline for receiving input data, directly performing a mathematical function on the input data in order to generate output data, and storing the output data in memory;

(b) wherein the mathematical function is directly performed in the computer graphics pipeline.

33. A system for executing a function in a computer graphics pipeline, comprising:

(a) means for receiving input data in a computer graphics pipeline;

(b) means for directly performing a mathematical function on the input data in order to generate output data, wherein the mathematical function is directly performed in the computer graphics pipeline; and

(c) means for storing the output data in memory on the computer graphics pipeline.

34. A method for executing a function, comprising:

(a) receiving input data;

(b) identifying the function to be executed on the input data;

(c) pre-processing the input data based on the function to be executed on the input data;

(d) processing the input data utilizing a plurality of operations independent of the function to be executed on the input data;

(e) post-processing the input data to generate output data; and

(f) storing the output data in memory.

35. The method as recited in claim 34, wherein the pre-processing includes adding a one (1) to the phase of the input data if the function to be executed on the input data is cosine.

36. The method as recited in claim 34, wherein the pre-processing includes multiplying the input data by $(1/(2\pi) + 1)$ if the function to be executed on the input data is at least one of sine and cosine.

37. The method as recited in claim 34, wherein the pre-processing includes performing a conditional 1's complement operation on the input data if the function to be executed on the input data is at least one of sine or cosine.

38. The method as recited in claim 34, wherein the pre-processing includes performing a barrel shift operation on the input data if the function to be executed on the input data is the exponent operation.

39. The method as recited in claim 34, wherein the processing includes extracting a set of most significant bits and a set of least significant bits from a mantissa associated with the input data.

40. The method as recited in claim 34, wherein the processing further includes conditionally adding a one (1) to the most significant bits.

41. The method as recited in claim 34, wherein the processing includes calculating a Taylor Series.

42. The method as recited in claim 41, wherein the processing includes looking up information in a plurality of tables.

43. The method as recited in claim 42, wherein the tables are utilized based on the function to be executed on the input data.

44. The method as recited in claim 42, wherein the information includes a plurality of derivatives.

45. The method as recited in claim 43, wherein the tables are hard-coded.

46. The method as recited in claim 43, wherein the tables are programmable.

47. The method as recited in claim 43, wherein the tables are loaded at runtime.

48. The method as recited in claim 44, wherein the derivatives are summed.

49. The method as recited in claim 34, wherein (b)-(f) are carried out in one cycle.

50. The method as recited in claim 34, wherein the function includes at least one of sin(x) and cos(x), where x is in at least one of degrees and radians.

51. A computer program product for executing a function, comprising:
(a) computer code for receiving input data;
(b) computer code for identifying the function to be executed on the input data;
(c) computer code for pre-processing the input data based on the function to be executed on the input data;
(d) computer code for processing the input data utilizing a plurality of operations independent of the function to be executed on the input data;
(e) computer code for post-processing the input data to generate output data; and
(f) computer code for storing the output data in memory.

52. A system for executing a function, comprising:
(a) logic for receiving input data;
(b) logic for identifying the function to be executed on the input data;
(c) logic for pre-processing the input data based on the function to be executed on the input data;
(d) logic for processing the input data utilizing a plurality of operations independent of the function to be executed on the input data;
(e) logic for post-processing the input data to generate output data; and
(f) logic for storing the output data in memory.

53. A method for executing a function in a computer graphics pipeline, comprising:
(a) receiving input data in a computer graphics pipeline;
(b) identifying a sign, an exponent, and a mantissa associated with the input data utilizing the computer graphics pipeline;
(c) normalizing the input data utilizing the computer graphics pipeline;
(d) if the function includes an exponent function, executing a barrel shift operation on the input data utilizing the computer graphics pipeline;

(e) if the function includes a cosine function, adding a one (1) to the phase of the input data utilizing the computer graphics pipeline;

(f) if the function includes at least one of a sine function and a cosine function, multiplying the input data by $(1/(2\pi) + 1)$ and performing a conditional 1's complement operation on the input data utilizing the computer graphics pipeline;

(g) extracting a set of most significant bits and a set of least significant bits from the mantissa associated with the input data utilizing the computer graphics pipeline;

(h) adding a one (1) to the most significant bits utilizing the computer graphics pipeline;

(i) looking up a plurality of derivatives based on the most significant bits utilizing the computer graphics pipeline;

(j) calculating a Taylor Series with the derivatives and the least significant bits utilizing the computer graphics pipeline;

(k) post-processing the input data to generate output data utilizing the computer graphics pipeline;

(l) storing the output data in memory on the computer graphics pipeline.